Distributed Algorithms for Controlling Multiple Mobile Robots*

Kazuo Sugihara† Ichiro Suzuki‡

January 14, 1994

Abstract

We discuss a method for controlling a group of mobile robots in a distributed manner. The method is fully distributed in the sense that each robot plans its motion individually based upon a given goal of the group and the observed positions of other robots. We illustrate the method by showing how a large number of robots can form an approximation of a circle, a simple polygon or a line segment in the plane. We also show how the robots can distribute themselves nearly uniformly within a circle or a convex polygon in the plane. Finally, we show how the robots can be divided into two or more groups. It turns out that in many cases most robots execute an identical, simple algorithm. The performance of the method is demonstrated by simulation.

1 Introduction

Development of autonomous mobile robots has been one of the major research efforts of many institutions, due to their immediate applicability in a wide variety of tasks such as space missions, operations in hazardous environments, and military operations. Leading research programs on mobile robots and related topics include (1) The MIT Mobile Robot Lab [2] [3] [4] [5] [6] [7] [8] [21] [23] [24] [25] [34]; (2) The Shakey Robot of SRI [30] [31] [32] [33]; (3) The Rover Robot of CMU [27] [28] [29]; (4) The TO-Rover Robot of University of Tokyo [39]; (5) Multiple Autonomous Underwater

*A preliminary version of this paper was presented at the 5th IEEE International Symposium on Intelligent Control, September 1990, Philadelphia, Pennsylvania. This work was supported in part by the National Science Foundation under grants CCR-9004346 and IRI-9307506, the Office of Naval Research under grant N00014-94-1-0284, and the Graduate School, University of Wisconsin – Milwaukee.

†Department of Information and Computer Sciences, University of Hawaii at Manoa, 2565 The Mall, Honolulu, HI 96822, U.S.A. (808) 948-6049

‡Department of Electrical Engineering and Computer Science, University of Wisconsin – Milwaukee, P.O. Box 784, Milwaukee, WI 53201, U.S.A. (414) 229-5689 *Send all correspondence to this author.*

Vehicles of National Bureau of Standards [15]; and (6) Autonomous Land Vehicle of DARPA [1] [16] [18].

The main subject of this paper is the problem of controlling *multiple* mobile robots in a *distributed* manner. Controlling a *single* robot, including navigation and motion planning, has been one of the central issues of mobile robot research (see, for example, [9] [10] [14] [19] [22] [26] [27] [35] [43]). Although some investigations on the problem of controlling *multiple* mobile robots have been started (see, for example, [11] [12] [13] [15] [17] [23] [37] [41] [42] [44]), our knowledge on how the motion of many mobile robots can be coordinated efficiently to achieve a given goal is still limited at present. As will be explained next, we have identified a class of interesting tasks that requires explicit cooperation and motion coordination of a large number of mobile robots, for which a *distributed* control method can be effective.

Consider the following three problems of coordinating the motion of about 50 mobile robots in the plane.

1. The robots are required to form an approximation of a simple geometric object, such as a circle, a simple polygon or a line segment, in order to build a barricade or to surround a given object. The size of the geometric object to be formed may be given, but its exact final location is not very important.

2. The robots are required to distribute themselves nearly uniformly within an area of a particular shape, such as a circle or a convex polygon, in order to protect the area from possible attacks.

3. The robots must be "divided" into two or more groups of approximately equal sizes, where in the final state different groups must occupy distant portions of the plane.

A centralized solution for these problems may consist of the following steps: (1) Count the total number of robots, (2) obtain the current position of each robot, (3) determine a final position for each robot, (4) generate a schedule for moving the robots from their current positions to their respective final positions, (5) broadcast the schedule to the robots, and (6) let the robots start executing the schedule simultaneously. Another centralized solution would be to let the robots communicate regularly and maintain the same information at all times, so that each robot can generate a plan for the entire set of robots by using the same information and then execute the portion of the plan applicable to itself [15]. It is conceivable, however, that such solutions may not always be desirable for the following reasons.

1. If the number of robots is large, the communication overhead required for collecting all relevant information may be prohibitively large. Also, it may be computationally infeasible to generate a schedule for the entire set of robots in real time.

2. A new schedule for the entire set of robots may have to be generated every time a robot becomes faulty.

Furthermore, the centralized solution can be considered to be rather "unnatural," since an exact final position of every robot must be determined before a schedule is generated, whereas none of the problems stated above requires that any specific robot be moved to any specific position.

The purpose of this paper is to present simple, fully distributed algorithms for the three problems stated above, assuming that each robot has a sensor for determining the positions of other robots. The basic idea is to let each robot execute a simple algorithm and generate its own schedule adaptively based upon the given goal and the positions of certain other robots. No explicit communication among the robots is necessary, and usually most of the robots execute the same algorithm. The algorithms are designed to work satisfactorily even if (1) the total number of robots is unknown and (2) a small number of robots become faulty during the operation, provided that nonfaulty robots can determine which robots are faulty. (The second condition is not always true for real robots.) Therefore, our solutions do not have the drawbacks of centralized solutions mentioned in the previous paragraph.

The idea of distributively controlling robots outlined above has interested a number of researchers in the last several years. The cellular robotic systems of Wang and Beni [42] consists of a large number of robots that operate in a cellular space under distributed control. They consider the problem of generating certain 1- and 2-dimensional cellular patterns using distributed control, and discuss interesting applications of the technique to the design of sensor arrays and escape systems. Another example is the dynamically reconfigurable robotic system of Fukuda and Nakagawa [13]. This system consists of many simple cells that can detach and combine autonomously to change its overall shape depending on the task and environment. Fujimura [12] investigates how the planning algorithms, knowledge about the environment and action intervals of robots affect the overall performance of the group of robots, and reports interesting simulation results for the case of two robots moving toward their respective goal positions avoiding collision. Mataric studies how local interactions of artificial agents can be designed to produce emergent group behaviors [23], and how different levels of intelligence of individual robots affect the group dynamics. Parker [34] discusses an architecture for adaptive action selection in a team of cooperative agents, and applies it to the example of a group of janitorial service robots. The results reported in this paper are based on the work done independently by the authors in 1988 and subsequently published in a preliminary form in 1990 [37]. To our knowledge, the particular problems addressed here have not been investigated before.

As is commonly done in many of the papers mentioned in the preceding paragraph, in this paper we use simulation for evaluating the performance of the proposed algorithms, representing each robot as a disc and employing a simple collision avoidance strategy. Also, we simulate sensor and control errors, and briefly discuss its effects on the performance of the algorithms. The problems considered here require only approximate solutions, and in this regard, the simulation results we obtained seem to indicate that the algorithms perform as expected (though at least theoretically, some of the algorithms can fail if the initial distribution of the robots has certain symmetry [illegible] egularity and there are no sensor and control errors). Formal analysis of the per[illegible]nce of the proposed algorithms, in terms of the initial distribution of the robot[illegible]peed of convergence and the quality of the resulting distribution, is left for future research.

The rest of this paper is organized as follows. We state the basic assumptions in Section 2, and then describe the collision avoidance strategy in Section 3. The problem of forming an approximation of a circle is discussed in Section 4. Section 5 illustrates a method for forming an n-sided simple polygon, where n is much smaller than the total number of robots. The problem of distributing robots nearly uniformly within a circle or a convex polygon is discussed in Section 6. A method for forming a line segment is also presented. In Section 7 we present a simple solution for the problem of dividing robo[illegible] two or more groups. Concluding remarks are found in Section 8.

2 Assumptions

For the purpose of simulation, we represent a robot as a disc that is 40 centimeters in diameter [1] and assume the following.

1. A robot can monitor the positions of other robots and move in any desired direction at any speed not exceeding the given maximum of 5 centimeters per second.
2. Initially, the robots can be distributed in a circular region of up to 20 meters in diameter.
3. The robots do not have a common x-y coordinate system. (This implies, for example, that we cannot program a robot to "move to position $(0,0)$" or "move north," expecting that they will all move toward a single point or in the same direction, respectively.)
4. For any robot, all other robots "look" identical. (This implies, for example, that we cannot program a robot to "find a red robot and move toward it.")

We note that the fourth assumption does not imply that a human operator cannot distinguish different robots. For example, occasionally we may need a small number of selected robots to behave differently from the rest of the robots. In such cases, the human operator may wish to control these robots manually while the rest of the robots are executing the given algorithm. It would of course be highly desirable if no human intervention is needed to solve the given problem, but apparently that is not always the case. (For example, leader election and symmetry breaking are not always

[1]The MIT AI Lab mobile robot [2] is cylindrical and is about 17 inches in diameter. The robot used in [24] is 30 centimeters in diameter. The robot used in [25] is not cylindrical, but is 12 inches long.

possible, under certain assumptions [38].) So in this paper we discuss both types of algorithms, those that do not require any human intervention before termination, and those that use such intervention during execution as useful guidelines.

Unless otherwise stated, the following are assumed regarding sensor and control errors in the simulation presented below.

1. The direction of and disance to another robot that a robot measures can be inaccurate by up to 5° and 5%, respectively.

2. If a robot wants to move in a certain direction at a certain speed, then its actual direction of move and speed can be inaccurate by up to 5° and 5%, respectively.

(Actually, we found that in many cases, the behavior of the algorithms we present does not change significantly even if the inaccuracies are assumed to be up to 10° and 10%, respectively.) The distribution of errors is assumed to be uniform. The simulation results we present in the subsequent sections are some of the typical ones we obtained out of a large number of runs for different numbers of robots (from 20 to 50) and their initial distributions.[2]

3 Collision Avoidance

We have developed a simple collision avoidance strategy for the robots, and used it in the simulation of all the robot algorithms we present. The strategy is that if a robot detects another robot within distance 20 centimeters (measured between the surfaces of the robots) in the direction of its move, then it swerves to left, minimally, provided that it successfully finds a direction that is clear of any such robot. If no such left swerve (including a backward move in an extreme case) is found to be possible, then the robot decides not to move until either its path becomes clear or a suitable left swerve becomes possible. One trick we use is that when a robot swerves, it slows down to one half the normal speed. We found this technique to be very effective in avoiding a robot from being "pushed" too far by a robot moving to the left immediately in front of it. (The method used in [25] involves stopping a robot for some fixed period, when it detects another robot in its way.)

Figure 1 shows how nine robots, each moving in the given fixed direction (subject to sensor and control errors), avoid collision using the strategy described above. Here, and in the rest of this paper, the hollow circles represent the positions of the robots every 5 seconds, and the black dots represent their final positions. The size of the circles and dots is in scale with the distances among the robots.

We found this strategy for collision avoidance to be effective for our purposes, and the robot algorithms we developed performed well in simulation when used with

[2]Reference [25] deals with 5 to 20 robots. Reference [36] discusses the behavior of 100 "agents" and mentions that there are other situations in which a much larger number of agents are needed to demonstrate phenomena of interest. The number of robots (20 to 50) we consider seems reasonable, i.e., large enough for the robots to consistently exhibit the behavior we expected, and small enough for us to actually build that many robots.

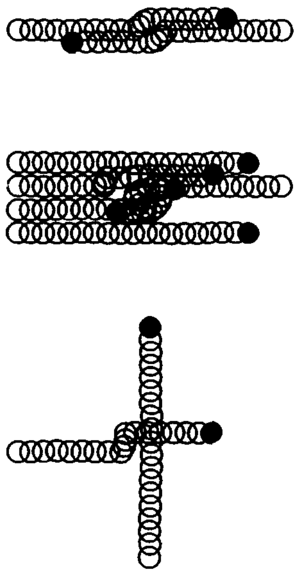

Figure 1: Example of collision avoidance.

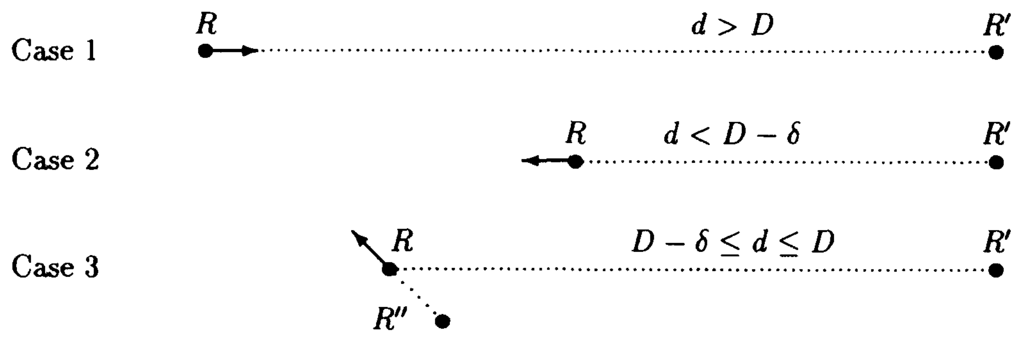

Figure 2: Algorithm CIRCLE.

it. Of course, as is the case with any "local" strategy for collision avoidance, there can be a situation in which a robot fails to find its path successfully. For example, a robot that enters an "U-shaped" region surrounded by stationary robots while moving towards its destination will never exit from that region. (Such a situation does not occur in our problem setting.) As is observed in [25], due to sensor errors and other uncertainties in real robot behavior, we believe that it is unlikely that this strategy causes other undesirable problems, such as deadlocks and infinite oscillations.

4 Forming an Approximation of a Circle

In the following we denote by N the total number of robots. As we stated before, the simulation has been done for various values of N in the range of 20 to 50.

4.1 Algorithm CIRCLE

The objective is to move the robots so that they will form an approximation of a circle having a given diameter D. Consider the following algorithm CIRCLE that will be executed by each robot R, where $\delta > 0$ is a small constant.

Algorithm CIRCLE

Robot R continuously monitors the positions of a farthest robot R' and a nearest robot R'', breaking ties arbitrarily, and moves as follows in real time, where d is the distance between R and R' (Figure 2):

Case 1 If $d > D$, then R moves toward R'.

Case 2 If $d < D - \delta$, then R moves away from R'.

Case 3 If $D - \delta \leq d \leq D$, then R moves away from R''.

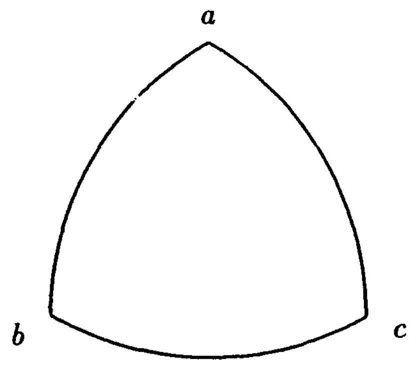

Figure 3: Reuleaux's triangle.

Case 3 of CIRCLE has the effect of distributing the robots more uniformly. Cases 1 and 2 will ensure that the distance between any robot and a robot farthest from it will be approximately D, so that the width of the resulting distribution of the robots will be D in any direction. This should certainly be true if N is large and the N robots are distributed nearly uniformly on the circumference of a circle having diameter D, but clearly this condition alone is not strong enough to ensure that the resulting distribution will be a circle. In fact, a convex figure whose width is a constant D in any direction is called an ***oval of constant width D*** [20], and a circle having diameter D is only an example of such a figure. Figure 3 shows another oval of constant width D, called ***Reuleaux's triangle***, obtained by drawing arcs $\overset{\frown}{ab}$, $\overset{\frown}{bc}$ and $\overset{\frown}{ca}$, with radii equal to D, from the vertices c, a and b, respectively, of an equilateral triangle abc with sides equal to D.

4.2 Simulation

Figure 4 shows a simulation result for 30 robots ($N = 30$) executing CIRCLE for 200 seconds. The values of D and δ are 10 meters and 10 centimeters, respectively. As we already stated, the maximum speed of a robot is 5 centimeters per second. The initial positions of the robots are generated randomly within a circle that is 20 meters in diameter. We have approximated the actual execution of CIRCLE by having each robot adjust its direction of move every 0.5 seconds. Figure 5 shows another result for 30 robots starting from a different initial distribution. In this result, we note that that the distribution of the robots has become somewhat similar to Reuleaux's triangle shown in Figure 3. Through additional simulation, we have observed that if the number of robots N is sufficiently large (say at least 20) and initially the robots are distributed randomly, then the resulting distribution of the robots is quite likely to be a fairly good approximation of either a circle or Reuleaux's triangle. (Some techniques for avoiding the formation of Reuleaux's triangle are discussed in [40].)

At least theoretically, the initial distribution of the robots is important. For example, if there are no sensor and control errors and the robots are arranged perfectly on a line, then the robots executing CIRCLE will always stay on the same line. Of

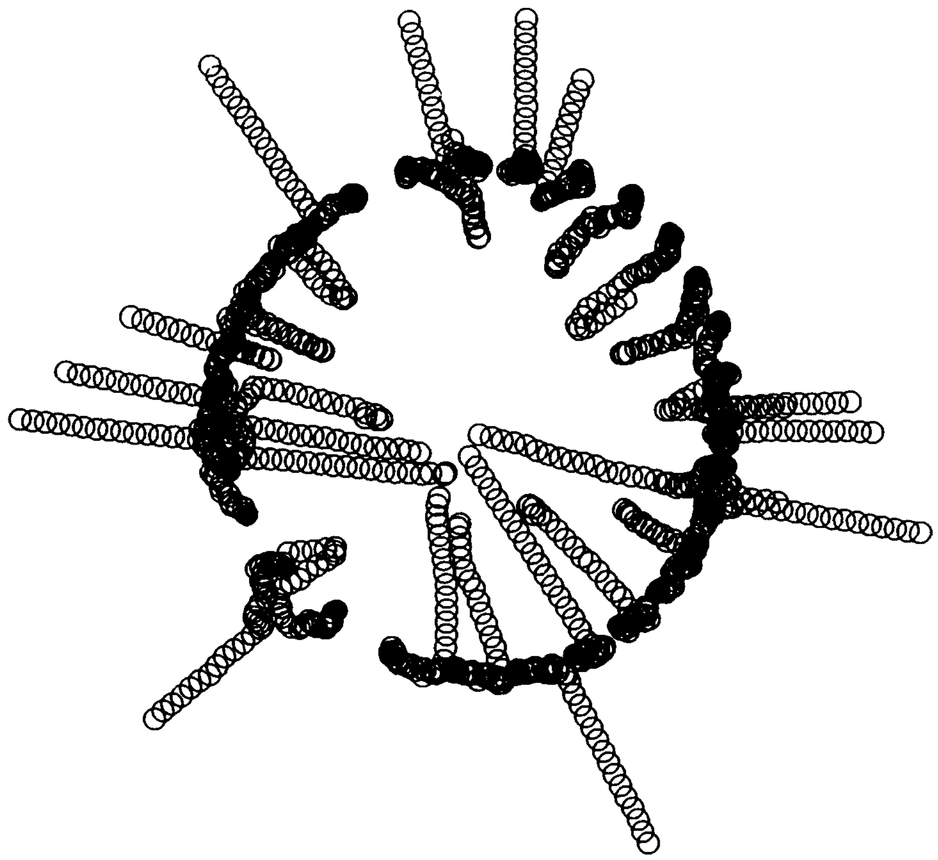

Figure 4: Execution of CIRCLE for 200 seconds.

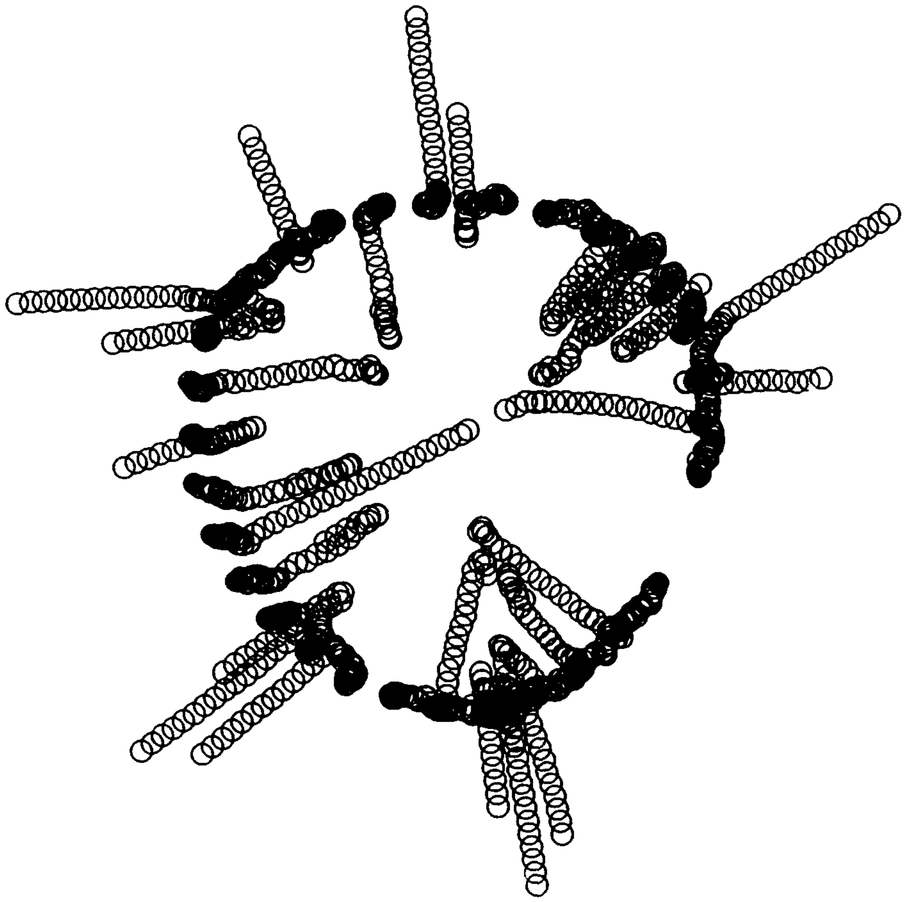

Figure 5: Execution of CIRCLE for 200 seconds, starting from another initial distribution.

course, an occurrence of such a situation is very unlikely for real robots. Moreover, we can easily destroy, with a high probability, any symmetry or regularity in the initial distribution by moving the robots randomly for a short period of time. Similar remarks apply to some of the algorithms presented in the rest of the paper.

4.3 Termination of the algorithm

The algorithm does not have any built-in mechanism for terminating its execution. Our position is that, when the human commander observes that a sufficiently good approximation of a circle has been generated, he can do either one of the following:

1. Stop all robots by broadcasting a signal.
2. Stop any *one* robot and let other robots continue to execute CIRCLE.

It can be verified by simulation that the second method of stopping only one robot will not destroy the distribution of the robots. Furthermore, if we control a single robot manually and move away from other robots, then the rest of the robots, still executing CIRCLE, will move (more or less) in the same direction maintaining the distribution. Although a fully automatic operation of the robots is highly desirable, some human intervention might still be necessary for many cases. So we do not necessarily consider this manual operation as an unacceptable drawback. Also, we believe that it should be quite feasible to control a small number of robot manually.

5 Forming a Simple Polygon

In this section we present a method for forming a simple n-sided polygon, where $n \geq 3$ is much smaller than the total number of robots N.

First, we let the robots execute algorithm CIRCLE until the resulting distribution approximates a circle sufficiently well so that each robot R can recognize its immediate "left" and "right" neighbors, denoted $l(R)$ and $r(R)$, respectively. The neighbors $l(R)$ and $r(R)$ are determined only once, and one way to select them is the following. Each robot R regards the nearest robot as $l(R)$, and $r(R)$ is the nearest robot in the half plane not containing $l(R)$ determined by the line passing through R perpendicular to the segment $\overline{R\,l(R)}$ (Figure 6). (The robots need not have a global sense of orientation, that is, "left" need not mean the same to all the robots.)

When $l(R)$ and $r(R)$ are determined for each robot R, we select n robots that will be the vertices of the n-sided polygon to be generated. Then we do the following, possibly concurrently.

1. Move the n robots manually to their desired final positions, in such a way that the order in which the n robots appear in the polygon will be the same as in the current approximation of a circle.
2. Let all other robots R execute the following algorithm CONTRACTION.

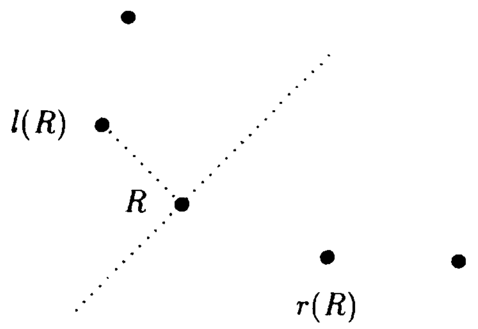

Figure 6: Choice of $l(R)$ and $r(R)$.

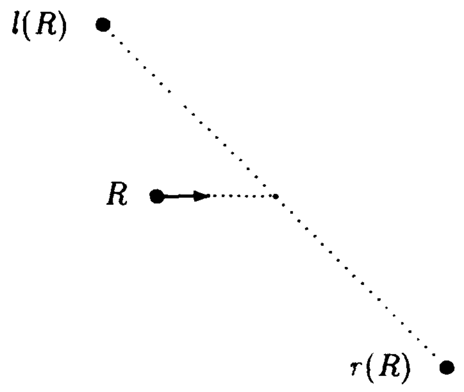

Figure 7: Algorithm CONTRACTION.

Algorithm CONTRACTION

Robot R continuously monitors the positions of $l(R)$ and $r(R)$, and moves toward the midpoint of the segment $\overline{l(R)\,r(R)}$, in real time (Figure 7).

Intuitively, algorithm CONTRACTION has the effect of transforming any curve formed by a sequence of robots between two robots that are *not* executing CONTRACTION into a line segment in which the robots are distributed nearly uniformly. Note that a robot executing CONTRACTION need not "know" which robots are *not* executing CONTRACTION.

Figure 8 shows the positions of 30 robots obtained by applying this method for 100 seconds, starting from a distribution reached by executing algorithm CIRCLE. The four robots that are at the vertices of the resulting polygon were moved manually.

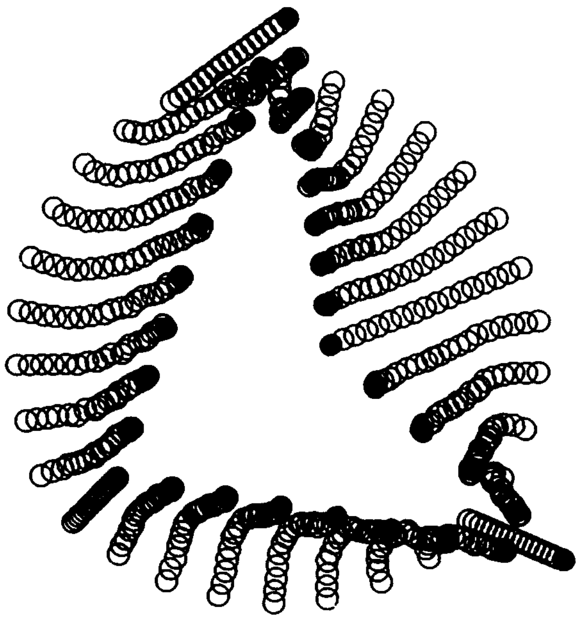

Figure 8: Formation of a polygon by CONTRACTION.

6 Distributing Robots within a Simple Geometric Object

6.1 Distribution within a circle

Robots can be distributed nearly uniformly within (an approximation of) a circle having diameter D if each robot R executes the following algorithm FILLCIRCLE.

Algorithm FILLCIRCLE

Robot R continuously monitors the positions of a farthest robot R' and a nearest robot R'', breaking ties arbitrarily, and moves as follows in real time, where d is the distance between R and R':

Case 1 If $d > D$, then R moves toward R'.

Case 2 If $d \leq D$, then R moves away from R''.

Case 1 of FILLCIRCLE ensures that the distance between any two robots will be at most D. Case 2 has the effect of distributing the robots more uniformly. Note that R' and R'' for R are not fixed and may change in real time as the positions of robots change.

Figure 9 shows the positions of 30 robots executing algorithm FILLCIRCLE for 100 seconds starting from their initial positions generated randomly. We observe that the robots are distributed fairly uniformly within a circular region. (We found that the region tends to be an approximation of either a circle or Reuleaux's triangle.) As is the case with algorithm CIRCLE, we can move the robots in any direction without destroying the distribution by moving any one robot manually.

6.2 Distribution within a convex polygon

Distributing robots nearly uniformly within a convex polygon requires a different technique. First, we select n robots that will be the vertices of the n-sided convex polygon, where $n \geq 3$ is much smaller than N. Then we do the following, possibly concurrently.

1. Move the n robots manually to their desired final positions.

2. Let all other robots R execute the following algorithm FILLPOLYGON.

Algorithm FILLPOLYGON

Robot R continuously monitors the positions of other robots and moves as follows in real time (Figure 10).

Case 1 If, as seen from R, all other robots lie in a wedge whose apex angle is less than or equal to π, then R moves into the wedge along the bisector of the apex.

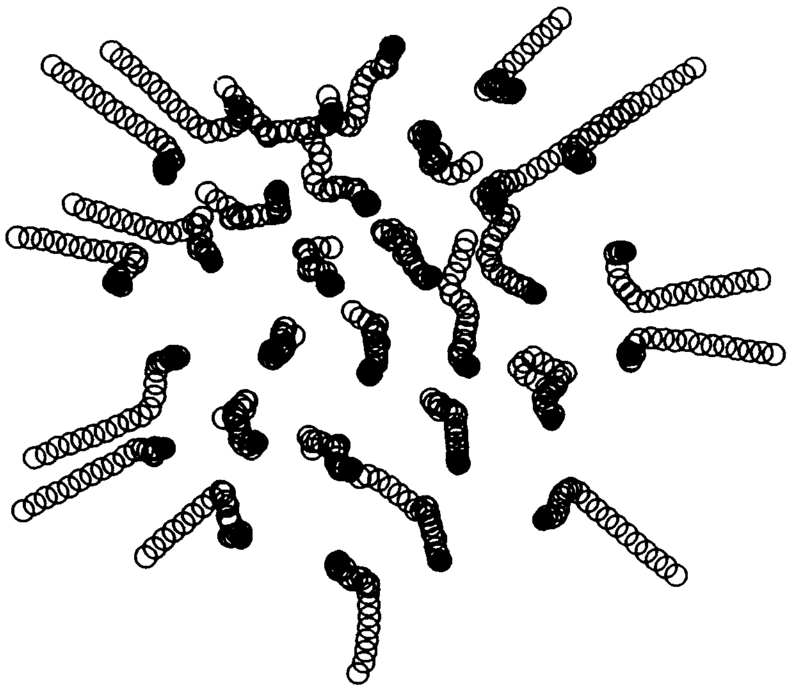

Figure 9: Execution of FILLCIRCLE for 100 seconds.

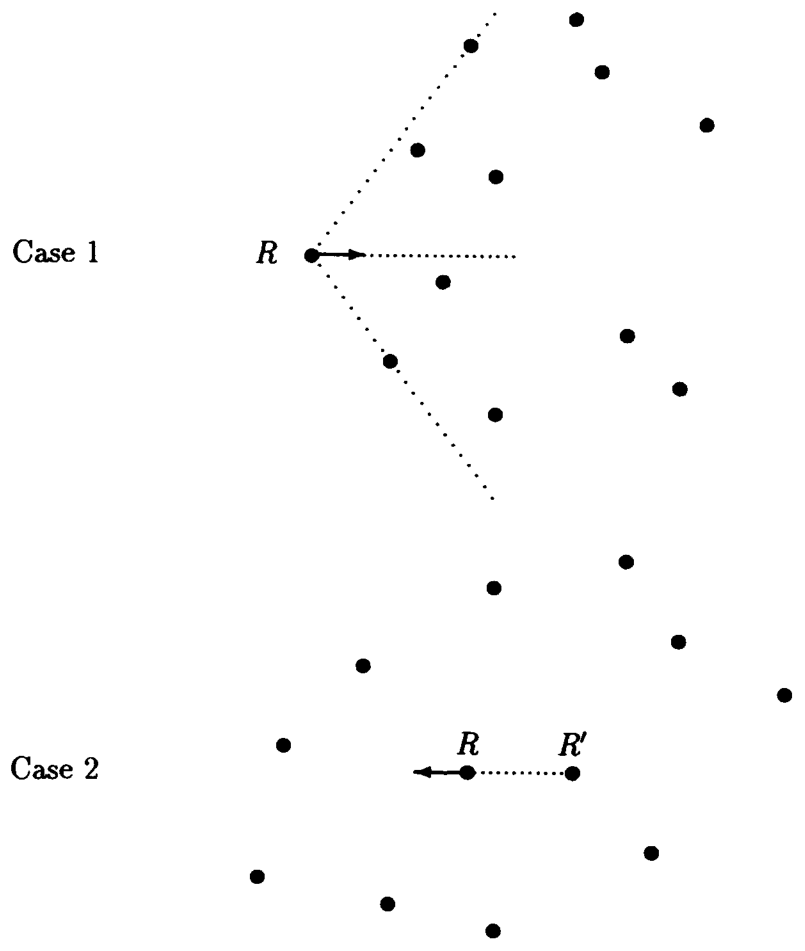

Figure 10: Algorithm FILLPOLYGON.

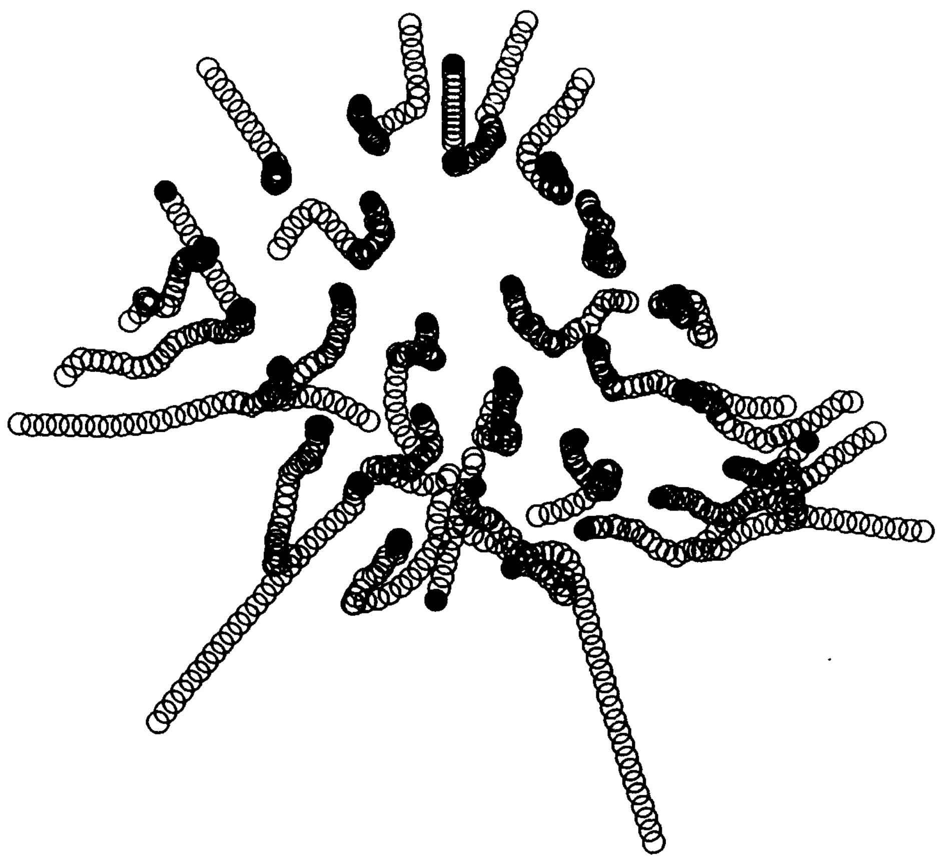

Figure 11: Distributing robots within a convex polygon by FILLPOLYGON.

Case 2 Otherwise, R moves away from a nearest robot, breaking ties arbitrarily.

Case 1 has the effect of moving a robot into the convex polygon determined by the selected n robots. Case 2 is expected to distribute the robots more uniformly within the polygon. As is the case with CONTRACTION, a robot executing FILLPOLYGON need not "know" which robots are *not* executing FILLPOLYGON.

Figure 11 shows the positions of 30 robots obtained by applying this method for 150 seconds starting from the initial distribution shown in Figure 4. The four robots that are at the vertices of the resulting convex polygon were moved manually. We observe that the robots are distributed fairly uniformly within the polygon, except that there is a somewhat convex region near the upper corner containing no robots. Apparently, none of the robots surrounding the region can enter the region, by Case 2 of Figure 10. In general, if the number of robots is very large, it is possible to have many such "bubbles" inside the polygon (and also in the circular region of Subsection 6.1). We might be able to reduce the possibility of having such bubbles by allowing the robots to move more randomly within the polygon.

As before, by controlling a small number of robots manually, we can (1) move the robots in any direction without destroying the distribution and (2) transform the polygon into another convex polygon.

6.3 Formation of a line segment

This is similar to the problem of distributing robots within a convex polygon discussed in the previous subsection. We select *two* robots that will be the endpoints of the line segment and do the following, possibly concurrently.

1. Move the two robots manually to their desired final positions.

2. Let all other robots execute algorithm FILLPOLYGON.

Figure 12 shows the positions of 20 robots obtained by applying this method for 200 seconds starting from their initial positions generated randomly. The two robots that are at the endpoints of the resulting segment were moved manually. We see that the final distribution of the robots is a slight zig-zag formation, that might still be considered a sufficiently reasonable approximation of a line segment.

For this simulation only, we assumed that the sensor inaccuracies regarding direction are up to 1°, rather than 5°. (The control inaccuracies regarding direction and inaccuracies regarding distance are still up to 5° and 5%, respectively.) Through additional simulation, we found that the algorithm is sensitive to directional sensor errors: In the presence of even up to 2° of such errors, the robots tend to form two chains, each connecting the given two endpoints, rather than a single line segment. The reason for this sensitivity is the following. When the convex hull of the robot positions becomes sufficiently "thin," then for a robot R for which the apex angle of the wedge in Case 1 of FILLPOLYGON (Figure 10) is less than or equal to π, the angle can erroneously be evaluated to be greater than π. Then the robot executes Case 2 instead of Case 1, and fails to make the convex hull even thinner.

The final distribution shown in Figure 12 can be converted to a more uniform distribution, by letting each robot R (except the two stationary ones) determine the neighbors $l(R)$ and $r(R)$ and execute algorithm CONTRACTION (as in Figure 7) starting from that distribution. Furthermore, the location and the length of the segment can be changed by moving the two robots at the endpoints of the segment manually while the rest of the robots are executing CONTRACTION. Figure 13 shows an example of this operation.

7 Dividing Robots into Groups

We present a method for dividing N robots into m groups of approximately equal sizes, where m is much smaller than N.

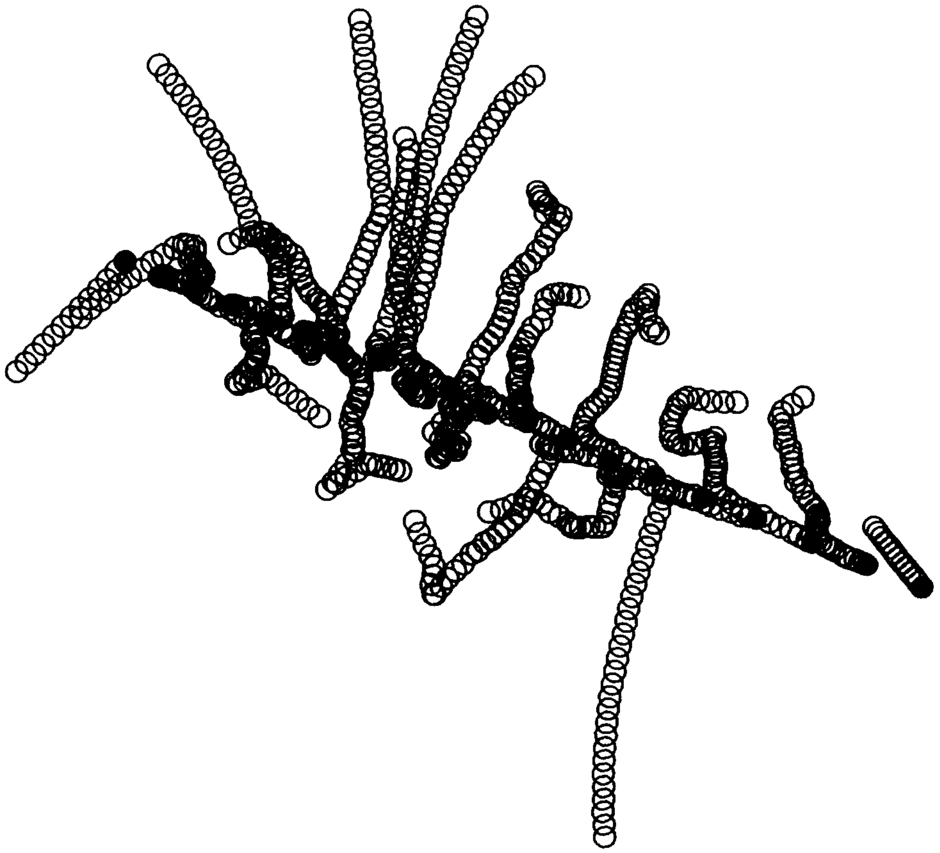

Figure 12: Formation of a line segment by FILLPOLYGON.

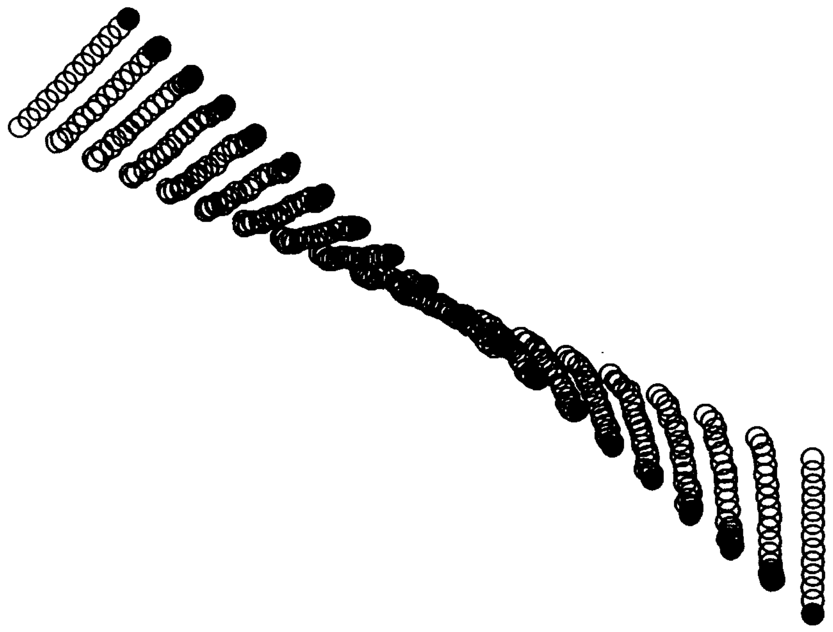

Figure 13: Changing the location and the length of a segment by CONTRACTION.

First, let the robots execute algorithm CIRCLE until $l(R)$ and $r(R)$ can be determined for each robot R, as in the case of forming a simple polygon discussed in Section 5. Next, select m robots that will be the "leaders" of the m groups. To make the sizes of the groups approximately the same, the m robots must be chosen so that they are nearly evenly distributed. Finally, we do the following, possibly concurrently.

1. Control the m robots manually and move them away from each other.
2. Let all other robots R execute algorithm FOLLOW given below.

Algorithm FOLLOW

Robot R stays in the current position until one of $l(R)$ and $r(R)$ moves, and then "follows" the one, call it R', that has started to move first (breaking a tie arbitrarily) as follows: R memorizes the initial distance d between R and R', continuously monitors the position of R', and moves, in real time, toward R' if the distance between R and R' becomes "sufficiently" greater than d (say, 10% greater than d including a safety margin, if there can be up to 5% sensor errors regarding distance).

Figure 14 shows the positions of 30 robots obtained by applying this method with $m = 3$ for 100 seconds, starting from a distribution obtained by CIRCLE. It is easy to see that the three groups can be moved to any distant locations by moving the three leaders manually.

8 Concluding Remarks

We have presented simple, fully distributed algorithms for certain problems of coordinating the motion of many mobile robots in the plane. The results of simulation seem to indicate that the algorithms we have obtained can be an attractive alternative to the conventional centralized control methods.

The following problems are suggested for future research: (1) To give formal analysis of the performance of the algorithms, and (2) to find complex tasks that can be performed efficiently by multiple mobile robots that are controlled distributively.

References

[1] J. A. Adam, "Aerospace and military," *IEEE Spectrum*, Vol. 23, No. 1, 1986, pp. 76–81.

[2] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robotics and Automation*, Vol. RA-2, No. 1, 1986, pp. 14–23.

[3] R. A. Brooks, "A hardware retargetable distributed layered architecture for mobile robot control," *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, 1987, pp. 106–110.

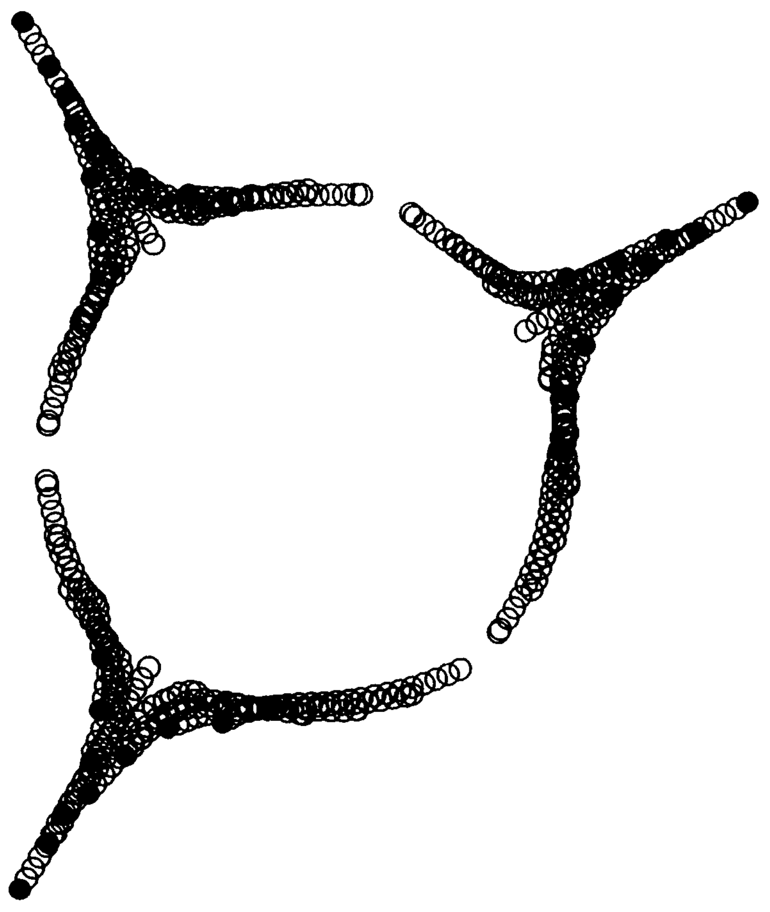
Figure 14: Dividing 30 robots into three groups.

[4] R. A. Brooks, "A robot that walks; Emergent behavior from carefully evolved network," *Neural Computation*, Vol. 1, No. 2, 1989, pp. 253–262.

[5] R. A. Brooks, "Intelligence without reason," *Proc. 12th Int. Conf. on Artificial Intelligence (IJCAI-91)*, 1991, pp. 569–595.

[6] R. A. Brooks and J. H. Connel, "Asynchronous distributed control system for a mobile robot," in *SPIE Cambridge Symp. Optical and Opto-Electronic Eng. Proc.*, Vol. 727, 1986, pp. 77-84.

[7] J. H. Connel, "Creature design with the subsumption architecture," *Proc. Int. Conf. on Artificial Intelligence (IJCAI-87)*, 1987, pp. 77-84.

[8] J. H. Connel, *Minimalist Mobile Robotics: A Colony Architecture for an Artificial Creature*, Academic Press, 1990.

[9] J. H. Connel, "SSS: A hybrid architecture applied to robot navigation," *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, France, 1992, pp. 2719–2724.

[10] J. L. Crowley, "Navigation for an intelligent mobile robot," *IEEE J. Robotics and Automation*, Vol. RA-1, No. 1, 1985, pp. 31–41.

[11] E. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Proc. IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, April 1986, pp. 1419–1424.

[12] K. Fujimura, "Model of reactive planning for multiple mobile agents," *Proc. IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, June 1991, pp. 1503–1509.

[13] T. Fukuda and S. Nakagawa, "Approach to the dynamically reconfigurable robot systems," *Journal of Intelligent and Robotics Systems*, Vol. 1, 1988, pp. 55-72.

[14] G. Giralt, R. Chatila and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," in *First Int. Symp. on Robotics Research*, M. Brady and R. Paul, Eds., 1983, The MIT Press.

[15] M. Herman and J. S. Albus, "Overview of the multiple autonomous underwater vehicles (MAUV) project," *Proc. IEEE Int. Conf. on Robotics and Automation*, Philadelphia, PA, April 1988, pp. 618–620.

[16] T. Kanade, C. Thorpe and W. Whittaker, "Autonomous land vehicle project at CMU," *Proc. 1986 ACM Computer Conference*, Cincinnati, OH, February 1986.

[17] K. Kant and S. W. Zucker, "Trajectory planning in time-varying environments, 1: TPP=PPP+VPP," Tech. Rep. TR-84-7R, Computer Vision and Robotics Laboratory, McGill University, Canada, November 1984.

[18] J. M. Lowrie, M. Thomas, K. Gremban and M. Turk, "The autonomous land vehicle (ALV) preliminary road-following demonstration," *Proc. SPIE Vol. 579 (Intelligent Robots and Computer Vision)*, September 1985, pp. 336–350.

[19] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision free paths among polyhedral obstacles," *Comm. ACM*, Vol. 22, No. 10, 1979, pp. 560–570.

[20] L. A. Lyusternik, *Convex Figures and Polyhedra*, D. C. Heath and Co., Boston, MA, 1966.

[21] M. J. Mataric, "Environmental learning using a distributed representation," *Proc. IEEE Int. Conf. on Robotics and Automation*, Cincinnati, OH, 1990, pp. 402–406.

[22] M. J. Mataric, "Navigating with a rat brain: A neurobiologically-inspired model for robot spatial representation," in *From Animals to Animats: Int. Conf. on Simulation of Adaptive Behavior*, The MIT Press, 1990, pp. 169–175.

[23] M. J. Mataric, "Designing emergent behaviors: From local interactions to collective intelligence," in *From Animals to Animats 2: Int. Conf. on Simulation of Adaptive Behavior*, The MIT Press, 1993, pp. 432–441.

[24] M. J. Mataric, "Integration of representation into goal-driven behavior-based robots," *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 3, 1992, pp. 304–312.

[25] M. J. Mataric, "Minimizing complexity in controlling a collection of mobile robots," *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, France, 1992, pp. 830–835.

[26] M. J. Mataric and R. Brooks, "Learning a distributed map representation based on navigation behaviors," in *Japan-U.S.A. Symp. on Flexible Automation*, Kyoto, Japan, 1990, pp. 499-506.

[27] H. P. Moravec, *Robot Rover Visual Navigation*, UMI Research Press, Ann Arbor, MI, 1981.

[28] H. P. Moravec, "Rover visual obstacle avoidance," *Proc. Int. Conf. on Artificial Intelligence (IJCAI-81)*, 785–790.

[29] H. P. Moravec, "The Stanford cart and the CMU rover," *Proc. of the IEEE*, Vol. 71, No. 7, 1983, pp. 872–884.

[30] J. H. Munson, "The SRI intelligent automaton program," *Proc. First National Symposium on Industrial Robots*, Chicago, IL, April 1970, pp. 113–117.

[31] N. J. Nilsson, "A mobile automaton: an application of artificial intelligence techniques," *Proc. IJCAI-1*, Washington, DC, 1969.

[32] D. Nitzan *et al.*, "Machine intelligence research applied to industrial automation," Tenth Report, SRI International, November 1980.

[33] D. Nitzan *et al.*, "Machine intelligence research applied to industrial automation," Eleventh Report, SRI International, November 1982.

[34] L. Parker, "Adaptive action selection for cooperative agent teams," in *From Animals to Animats 2: Int. Conf. on Simulation of Adaptive Behavior*, The MIT Press, 1993, pp. 442–450.

[35] J. T. Schwartz, M. Sharir and J. Hopcroft (Eds), *Planning, Geometry, and Complexity of Robot Motion*, Ablex Publishing Co., Norwood, NJ, 1987.

[36] Y. Shoham and M. Tennenholtz, "Emergent conventions in multi-agent systems: Initial experimental results and observations," *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'92)*, San Mateo, CA, 1992, pp. 225–231.

[37] K. Sugihara and I. Suzuki, "Distributed motion coordination of multiple mobile robots," invited paper, *Proc. 5th IEEE Int. Symp. on Intelligent Control*, Philadelphia, Pennsylvania, September 1990, pp. 138–143.

[38] I. Suzuki and M. Yamashita, "Formation and agreement problems for anonymous mobile robots," *Proc. 31th Annual Allerton Conf. on Communication, Control, and Computing*, University of Illinois, Urbana, Illinois, October 1993.

[39] M. Takano and G. Odawara, "Development of new type of mobile robot TO-ROVER," *Proc. 13th Int. Symp. on Industrial Robots 7*, Chicago, IL, April 1983, pp. 20-81–20-89.

[40] O. Tanaka, M. Yamashita and I. Suzuki, "A note on motion coordination of distributed autonomous robots," (in Japanese) *Proc. 1993 Joint Symp. on Electronics and Information*, Institute of Electronics, Information and Communication Engineers, Yamaguchi, Japan, October 1992.

[41] P. Tournassoud, "A strategy for obstacle avoidance and its application to multi-robot systems," *Proc. IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, April 1986, pp. 1224-1229.

[42] J. Wang and G. Beni, "Cellular robotic systems: Self-organizing robots and kinetic pattern generation," *Proc. IEEE Int. Workshop on Intelligent Robotics and Systems*, Tokyo, Japan, 1988, pp. 139–144.

[43] C. K. Yap, "Algorithmic motion planning," in *Advances in Robotics Vol. 1: Algorithmic and Geometric Issues*, J. T. Schwartz and C. K. Yap (Eds.), Laurence Erlbaum Pub., NJ, 1987, pp. 95–139.

[44] D.-Y. Yeung and G. A. Bekey, "A decentralized approach to the motion planning problem for multiple mobile robots," *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, March 1987, pp. 1779–1784.